CS M151B Homework 2
Baolinh Nguyen
UID: 104732121

2.24 [5] <§2.7> Suppose the program counter (PC) is set to 0x2000 0000. Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address as 0x4000 0000? Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to this same address?

$0x\ 4000\ 0000 - 0x\ 2000\ 0000 = 2^{28}$

The MIPS jump instruction is loaded into a 32 bit address. However, only 26 bits are used for the target so it cannot be used to jump to an address greater than 26 bit in a single instruction.

The branch-on-equal instruction cannot be used since it is an I-format instruction which only has 16 bit for the address.

CS M151B Homework 2
Baolinh Nguyen
UID: 104732121

2.26 Consider the following MIPS loop:

```
LOOP: slt  $t2, $0,  $t1
      beq  $t2, $0,  DONE
      subi $t1, $t1, 1
      addi $s2, $s2, 2
      j    LOOP
DONE:
```

2.26.1 [5] <§2.7> Assume that the register $t1 is initialized to the value 10. What is the value in register $s2 assuming $s2 is initially zero?

2.26.2 [5] <§2.7> For each of the loops above, write the equivalent C code routine. Assume that the registers $s1, $s2, $t1, and $t2 are integers A, B, i, and temp, respectively.

2.26.3 [5] <§2.7> For the loops written in MIPS assembly above, assume that the register $t1 is initialized to the value N. How many MIPS instructions are executed?

2.26.1

slt: $t2 = (0 < $t1) = (0 < 10) = 1
beq: $t2 = 1 == 0 X
subi: $t1 = $t1 - 1 = 9
addi: $s2 = $s2 + 2 = 0 + 2 = 2
slt: $t2 = 0 < $t1 = 0 < 9 = 1
beq: $t2 = 1 == 0 X
subi: $t1 = 8
addi: $s2 = 4
: repeat until $t2 = 0 & $s2 = 20

0	10	12	4
2	9	14	3
4	8	16	2
6	7	18	1
8	6	20	0
10	5		

2.26.3 $t1 = N.
The loop executed N times and when N = 0, only 2 instructions are executed after the loop is exited. So,

5(N) + 2 instructions are executed.

CS M151B Homework 2
Baolinh Nguyen
UID: 104732121

2.46 Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 500 million arithmetic instructions, 300 million load/store instructions, 100 million branch instructions.

2.46.1 [5] <§2.19> Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, and the cost of increasing the clock cycle time by only 10%. Is this a good design choice? Why?

2.46.1

$ET_0 = CT_0 \times CPI_0$

$= CT_0 \times ((500\times10^6\times1) + (300\times10^6\times10) + (100\times10^6\times3))$

$ET_1 = CT_1 \times CPI_1$

$= 1.1CT_0 \times ((0.75\times500\times10^6\times1) + (300\times10^6\times10) + (100\times10^6\times3))$

$ET_0 = 3800\times10^6\ CT_0$

$ET_1 = 4042.5\times10^6\ CT_0$.

This is a bad design choice since Execution Time actually increases afterwards.